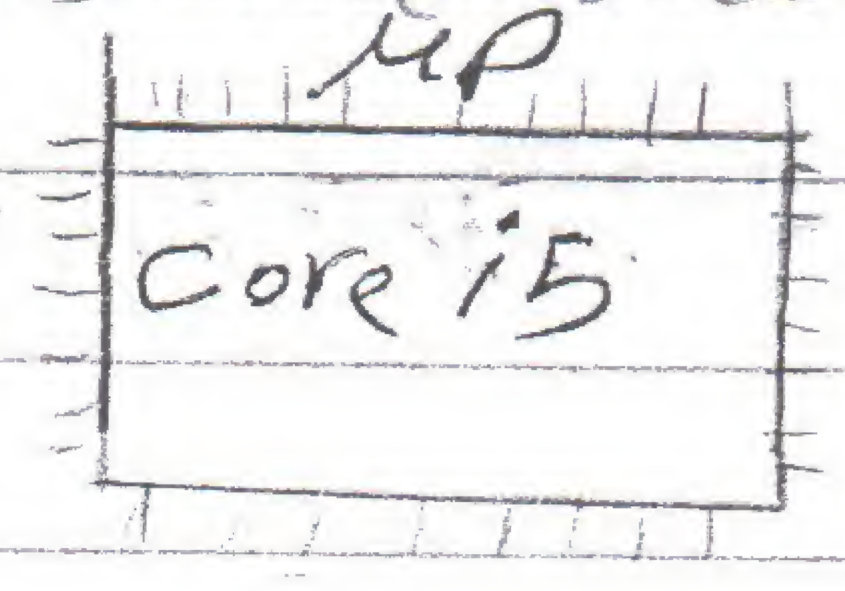
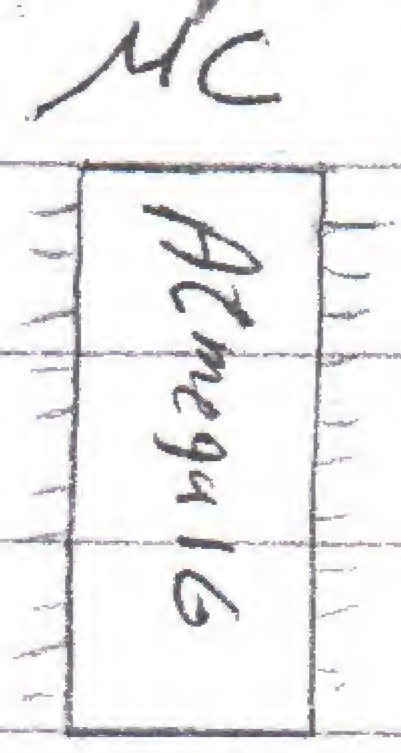
Introduction

what is Embedded system ?
it is soft ware on hardware to do cearrain perpose:

what is the defrent between μP and μC ?





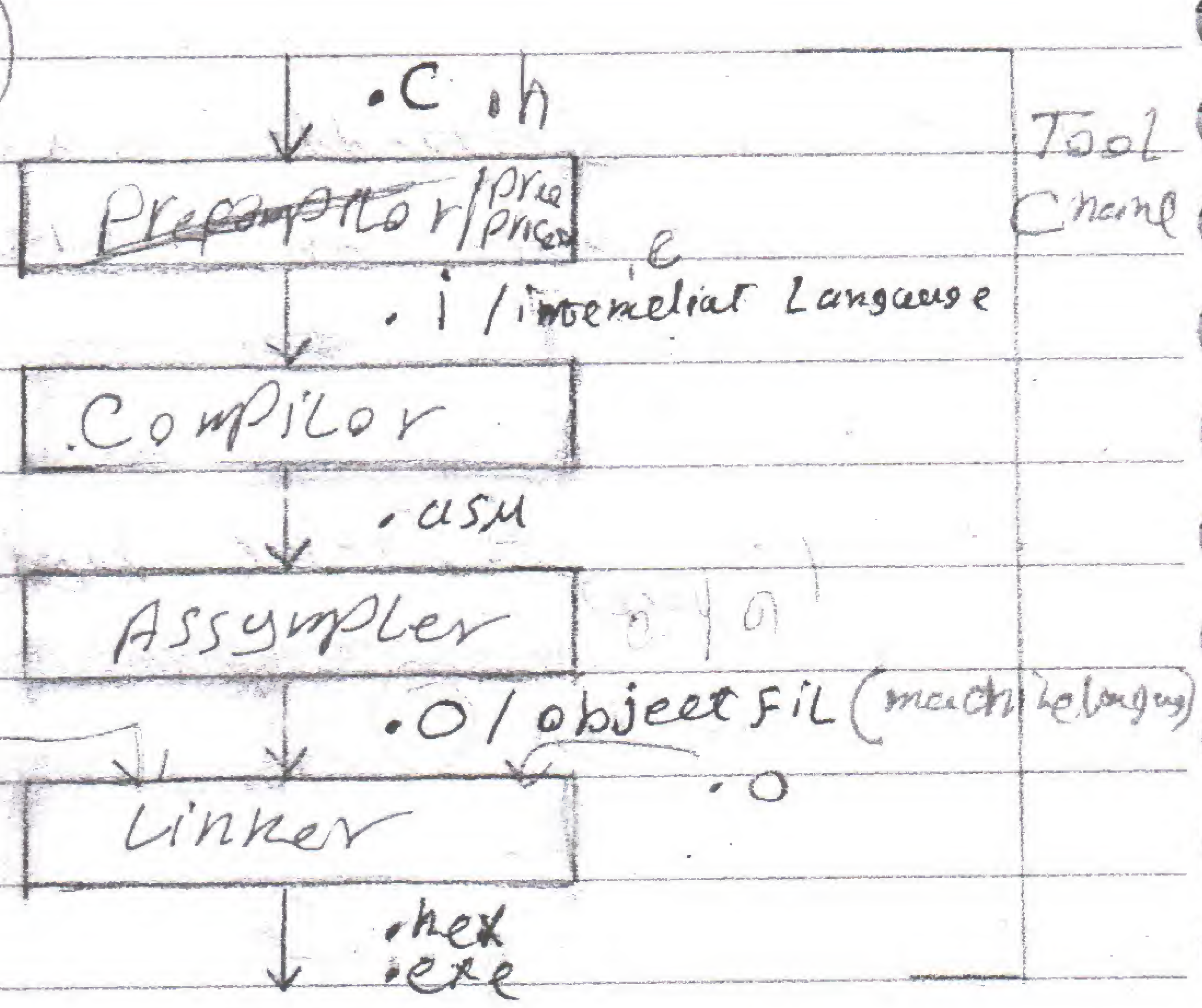
① μC has a μP in it.
② The μP in μC is very small and sheep.
③ μP like core i5 cost around 1000 L.E.
④ μC do one or some fixed functions.

S.W Constrans

① Cost
② optimized Code (memory) ذكر عطر ذكائك ٥
③ Timing (Soft real Time system, Hard realtime system)
④ Realiability. (Can be Trusted)
⑤ Power Consumption.

⑤ μP General Perpose , μc specific Perpose
⑥ μP dosent hase memory or Prephrals, μc hase
⑦
⑧
⑨

• Bild Process.



* كل Target ليها ال Tool Chane مختلفة عن التانية

* كلهم بيكتبو C (Embedd C)

وكلهم في الآخر بيحولوا ل machine Language

---

* Target: The H.w That what I'm develop for (Atmel)
* HOST: The machine I'm develop on it (labtop).
* Compilor: The Target and The host is The same.
* Cross compilor: Target diffrent from The host.

memory

Volatile (RAM) — Non Volatile (ROM) — Floating gate

• DRAM: Need refresh cycle • Flash: Code memory
• SRAM: Dosn't need refresh C. • EEPROM:

DRAM	SRAM	Flash	E²PROM
Power ↑	Power ↓	NAND	NOR
Cost ↓	Cost ↑	Cost ↑	Cost ↓
Speed ↓	Speed ↑	Speed ↑	Speed ↓
Capasitors	Transistors	لازم تمسح كلها	ممكن تمسح بايت Byte

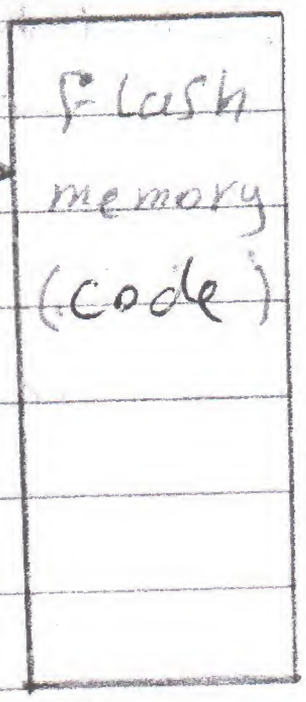
عند ما تفصل الكهرباء بتضيع ال Data

Excution Life cycle:

F	D	E	WB

① FETCH →

PC point To The instruction set and patit in (instruction Register)

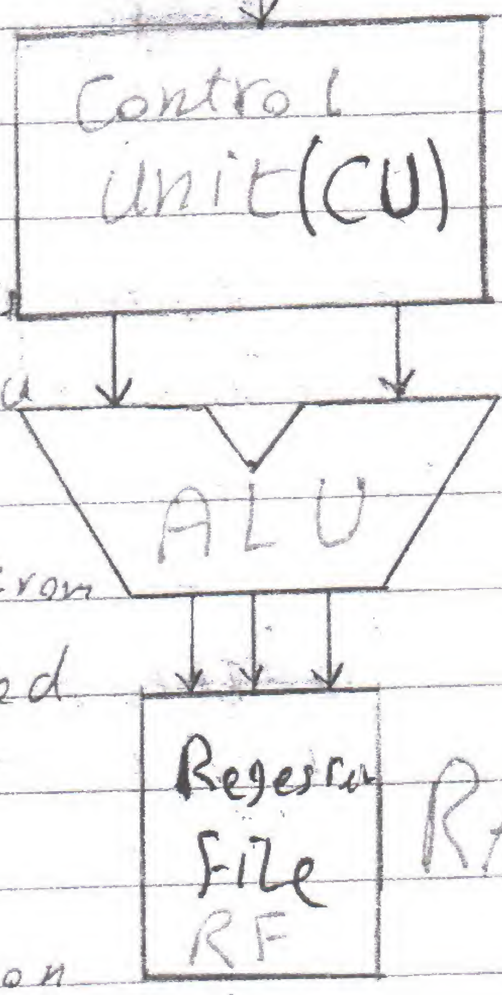


PC | Program Counter

IR | OPCode | Operands

② DeCode ↔

instruction register devide it to operation code and operands and send them seperately to ALU



③ excute →

ALU Take The Two Signals From The CU and excute The needed calculations

④ write Back →

Alu write The result Back on The RF

RAM

DATE 5/2/2016

- Registers:

(registers are memory locations.)

RF: Register file is a general purpose register near to ALU take fast

PC, IR: are aspical function registers.

RAM

GPR: General Perpose register
SFR: spical function register
SRAM: static Ram

- CU Takes (Sends) Two signals from ALU. That get them from IR was from PC.

PC point to the next instrution to be executed and put it in IR that devided them to two halfs the first is opcode and the second is operands. Then IR send the to CU sends two signls to ALU take the opcode and save it, operand writes them on RAM(RF) and then make the calculations and write them back in RF

C Language

High Level — Java, C++, C#

— C Language.

C is an intemediate language

— Assymply L.

Low Level — machine L.

Data Types

Primptive	Drived	user defined
int	② Array	Struct
Char	① Fanction	Union
Float	③ Pointer	Enum
double		
Void		

int size ⟶ 4 Byte

Char size ⟶ 1 Byte

float size ⟶ 4 Byte

double size ⟶ 8 Byte

machin dependent
native compiler
GCC
Linux

any code must have

① Preproccissor Directives — # include <stdio.h>

② Global Dedirations — int x, j

③ Main Fanction — int main( )

{

}

③ Operations

Arithmatic	Bitwise	Relational
+	&	<
-	|	<=
*	^	>
/	~	=>
%	<<	==
--	>>	!=
++		

④ Logical operators

① && ② || ③ !

⑤ Assigment operators

① += ② -= ③ *=

④ /= ⑤ %=

Left side of ASSign

X = X + 1
X += 1 same

[3] Code ex:3

write C program to compare IF (X != 10).

① Unary operator: ++, --

② Binary operator: +, -, <

③ Trinary operator: > ? True : Fals

Post inc, dec

```
int x = 10, y ;
y = x++ ;
printf("%d", x) ; → 11
printf("%d", y) ; → 10
```

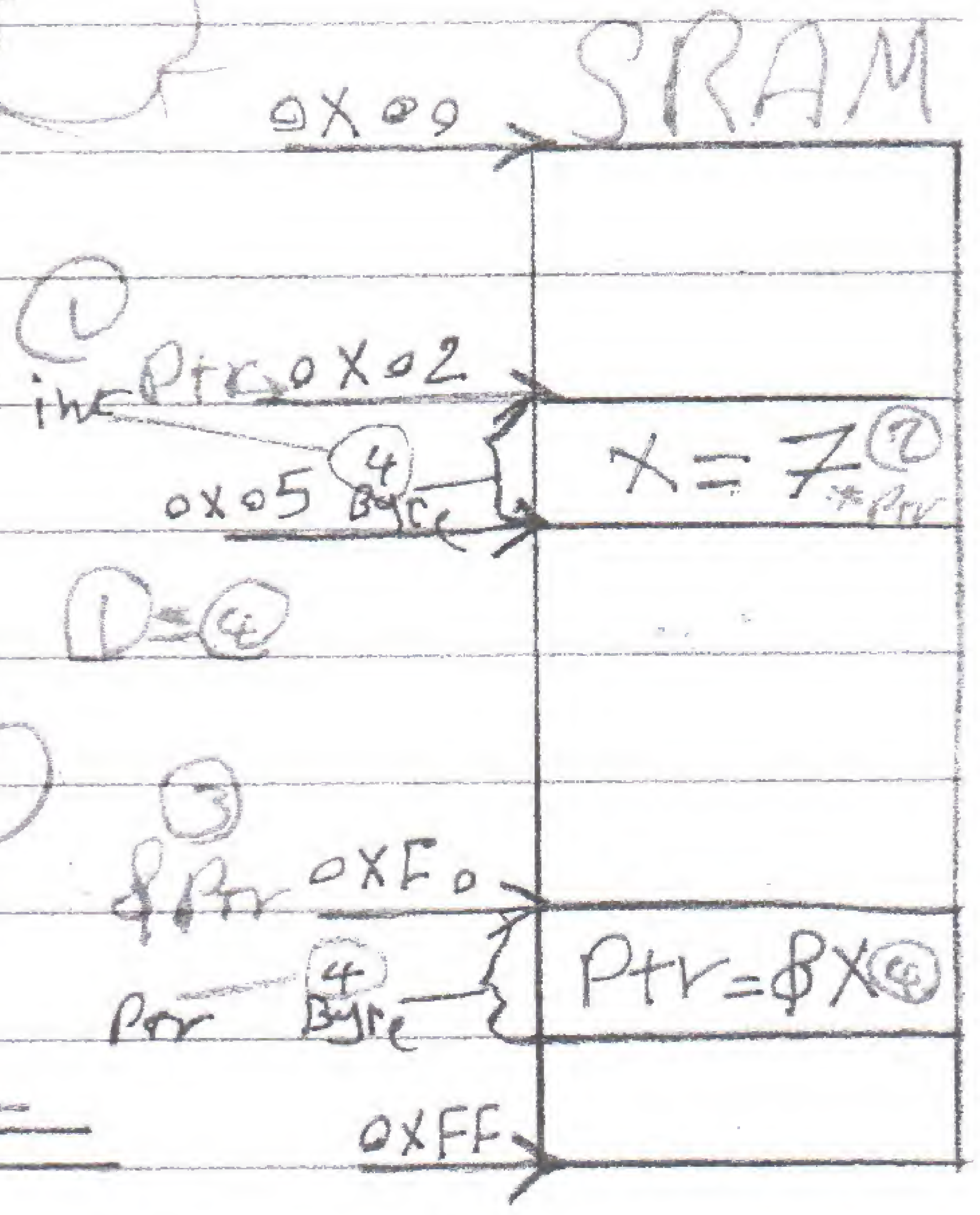
pree dec, inc

```
int x = 10, y ;
y = ++x ;
printf("%d", x) ; → 11
printf("%d", y) ; → 11
```

---

[1] Pointers

```
int *ptr ;
int (*) ptr = (&) X ;
*ptr = 7 ;
printf("%d", x) ; → 7
```



Size بتاع الـ Ptr ثابت ٤ بايت

---

② printf("%p", *ptr) ; → 7 (x)
①④ printf("%p", ptr) ; → 0X02 (&X)
③ printf("%p", &ptr) ; → 0XF0
② printf("%(p)", x) ; → 7

(p → hex.)

(N.B)

* The only defference between ptr to int and ptr to char is (memory STEP)

مع العناوين نعد

اللي ساكن في نفس الشارع، عناوينهم في البطاقة واقرب بس مساحات الشقة مختلفة

Constant Hacking:

Constant int X=10;
X=5; error

N.B Constr int *x; error

int *Ptr=&X;
*Ptr=3;
printf("%d",X); → ✓

(N.B) all data types are Singed by default.

unsigned $0 \longrightarrow 2^n - 1$
Signed $-2^{n-1} \longrightarrow 2^{n-1} - 1$

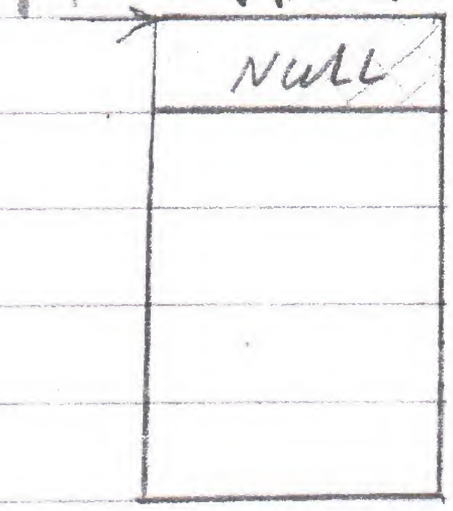
---

* كل الميزة في الـ Ptr انى معكم اغير بيه القيمة اللى
بيشاور عليها (*) ← البتاعة دى

Wild Pointer

int speed;
int *ptr; ← مينفعش
int *Ptr=Null; ← الحل

N.B



You can Create many Ptr for The same memory location.

Cod: ex4 writ c program can Add(or) Sum Two Variables. (scanf)

DATE 6/12/2016

Switch

```
switch (var)
    {
    case 1: ____;
        Break;
    case 2: ____;
        Break;
    default: ____;
        Break;
    }
```

4 Notes about switch

① Case grouping:

```
case 1:
case 2:
case 3:
    printf(" ");
    Break;
```

② Switch (~~Float, double~~)

switch only int and char.

③ You can't switch conditions.

PAGE
DATE

④ Unreachbul Code

```
switch(X)
{
    printf("  ");   <- unreachable
    scanf("  ");    <- unreachable
    case1:
        Break;
}
```

Maptable

```
switch(Y)
{
    Case1:
    Break;
    default;
    Break;
}
```

Flashmemory

Y	Addresses
1	0xFF0F
2	0xFA0C
3	0xA02A
defaut	0xFF0A

(N.B) Some case converted to IF, Else IF
على حسب اللى هيوفر ال execution time

① For Loop

For(initlization ; Condition ; itration action)

① initilization : ⟸ Code لازم يشوفه ال

② Condition : (itration كل مره (كل ال check عليه for هيعمل ال) هيشتغل على الأقل مرة

③ itration action : itration بيتنفذ من آخر كل

(N.B) for(i=0; x>i; i++) (;) ⟶ هيدخل كل اللوبس هنا
even if( ); while( ); زي الإثنين اللي ما الآخر كأنها شاملة الـ {}

N.B

you can make multi Like:

For (j=1, A=2, x=4 ; && || ; J++ i--)

② while

x=0;
while(i<10)
{

i++;
}

became For.Loop

Break;

① works with Looping(for, while)
② works with switch.

Continue

.makes Loop itration Skiped.
.once forLoop(for Ex) See Continue it will start over again The next itration.

for	while
بستخدم for لما اكون عارف عدد العمليات الى انا عاوزها	بستخدم while لما يكون factor خارجي ومش عارف هلف loop كام مرة

Infinit Loops

```
while(1)
{

}
```

```
for(jj)
{

}
```

---

③ do... while

```
do
{

} while(Condition);
```

For Best Practice

For(i=10; i>0; i--)

* في الـ Assymply بتتعرف في كل iteration instruction

جميلة

```
int i=0, J=10;
if( i++ && J++ )
{
}
```

Falls, True

S.C

printf("%d", i); → 1
printf("%d", J); → 10

```
int i=1, J=10
if( i++ || J++ )
{
}
```

printf("%d", i); → 2
printf("%d", J); → 10

12/12/2016

address[index]   ## 2 Arrays   Var[ ];

int x[10]; → Ruppesh.
int x[10] = {0}; → Zeros كلهم
int x[10] = {1,2,3,4,...}; هيحط الأماكن وهيعطي كل واحدة القيم بالترتيب والباقي مش initialize هيبقى zero.

5 Code Ex5:
write C program That init an array and print it with for loop.

Some notes of arrays

1. Name of array if a Constant pointer to The first element of array.
2. Size of array [index] must be defined.
3. arr[0] = x; ✓✓ تنفع
4. int arr[10]. → arr[0] — arr[9]
   arr[10] = 100; over accessing of an array
5. Arr[x]; ✗
6. int x; int y; ...; ✗ arr[10] = {x, y, ...}

Constant Pointer	Pointer to Constant
int x; (N.B Like array)	const int x;
int *const ptr = &x;	const int *ptr;
ptr++ error	ptr++ ✓
*ptr = 5; ✓	*ptr = 20 error
constant Pointer to an integer	Pointer to constant integer.

Address [Index]

int arr[100];

index = i

size

memory size

arr [0] → arr + (0 * 4) = arr first element

arr [1] → arr + 1 * 4 = arr+4

arr [2] → arr + 2 * 4 = arr + 8

arr [100] → arr + 4 * 4 = arr + 400

6 Code Exp:

write c Programe to scan elements of an array and scan X and compare return the number of element. if not find print "NF" on the screen. Flag F=-1;

Multi"D" Arrays

① 2D array

int Arr [2][3]

Rows = 2 (0, 1)

Colums = 3 (0, 1, 2)

0	arr[0][0]	arr[0][1]	arr[0][2]
1	arr[1][0]	arr[1][1]	arr[1][2]

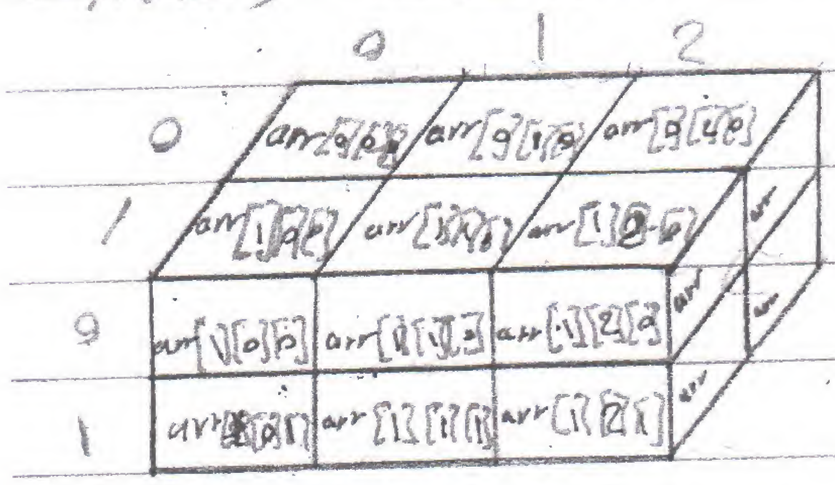
arr[0][0]
arr[0][1]
arr[0][2]
arr[1][0]
arr[1][1]
arr[1][2]

```
for (i=0; i<2; i++)
  { For (c=0; c<3; c++)
      { printf("%d", arr[i][c])
```

Code Ex 7:
write C program of 2D array and over access it.
Hint int arr[2][3];
arr[0][3] = 12; printf(arr[1][0]);

3D Array

int arr[2][3][2]
#Raws #Colums #



Use in handling 7-Sigmet display.

RAM

arr[0][0][0]
arr[0][1][0]
arr[0][2][0]
arr[1][0][0]
arr[1][1][0]
arr[1][2][0]
arr[0][0][1]
arr[0][1][1]
arr[0][1][2]
arr[1][0][1]
arr[1][1][1]
arr[1][2][1]

Pointer as an array

int X;
int *ptr = &X;
ptr[0] --> = *ptr
ptr[1]
ptr[2] / The only deffrent is array see in memory but Ptr No.

PAGE
DATE 13/2/2016

Functions

"Prototype"

returnType(opt)	Name	(list) Arguments

(Function declaration)

{

;

line of codes

Return;

}

Body (Function definition)

```
int sum (int x , int y)
{   int Resalt;
    Resalt= x + y;
    Return Resalt;
}
int main( )
{   int z;
    z= sum(5,10);
    printf("%d",z);
}
```

Code EX:

write C code to calcalate The Area of Cyrcle with Functions.

N.B

* والكتب الـ Body كله خيروبركة . كتبت الـ Prototype بس
  لازم تكون عامل الـ Definition في حتة تانية في نفس الـ file
* يفضل تعمل Definition بس multi declirate واحدة

Variable scope and Life time

```
int sum(int x, int y)
{
    printf("%d", x);
}
int main()
{
    int x, y;
    x = sum(5,10);
    y = sum(2,3);
}
```

RAM

x = 0
y = 2
x =
y =

Local

[9] Code Exp:
write C.code of 2 function one to sum and the second to sub. all with x, y

Global variables

File scope, function scope, (inner) Block scope.

وظيفة الـ Global Var هي حفظ البيانات في الذاكرة وبتت تروح تلقائية

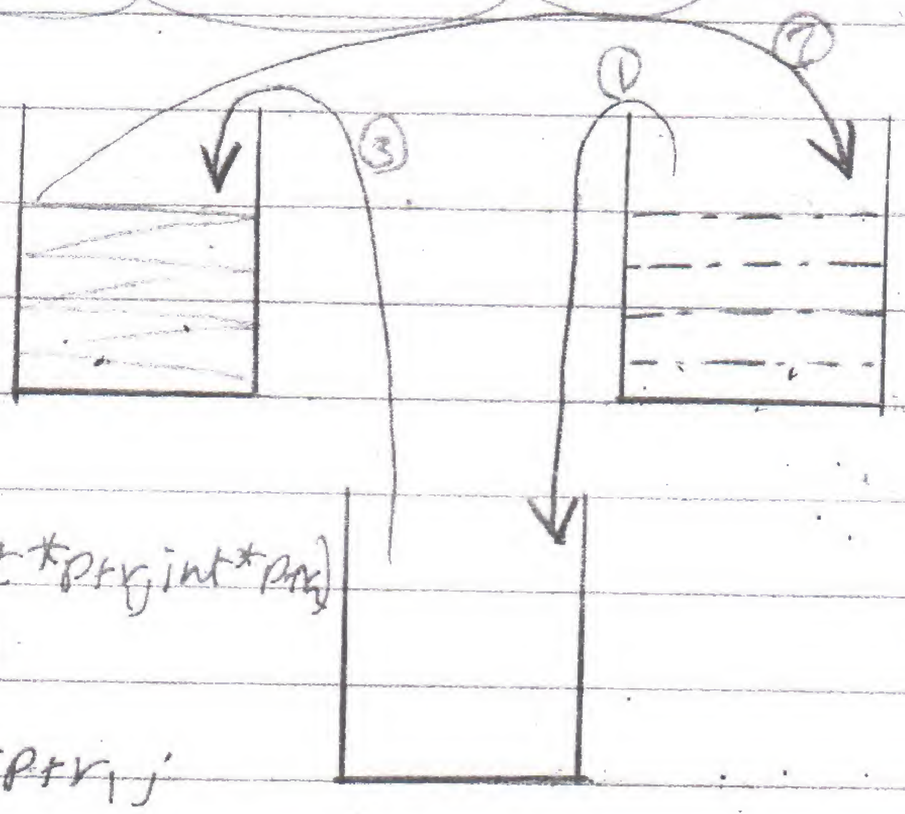
```
int x;
main()
{
}
```

Global Var

Life time	Scope
Program excution time	File scope

* لو عملت global و local بنفس الاسم متخافش
هيدي error و بيشوف الأقرب ليه (local)
* لو عملت كذا int x,x,x,x; global بنفس الاسم
متخافش هيدي error بس تعمل initialization لواحدة منهن

Swap Funetion




```
void swap(int *ptr1,int *ptr2)
{ int temp
  temp = *ptr1;
  *ptr1 = *ptr2;
  *ptr2 = temp;
}
```

Headers

#include "

swap → header ↔ prototype
C → definition.

#include "swap.h"

19/2/2016

Array to Function

```
void func(int *ptr, int size)
 {
   ______
 }

void main( )
 {   int arr[5];
     Func ( arr , 5 );
 }
```

* الـ Array بيتبعت
عبر By address

sizeof(arr) / sizeof(int)

10 Code Ex10:

write C code sends an array to func by address and print all of it.

Structure

مكتبات

```
Struct Student
  {  unsigned char class;
     int Id;
     char grade;
  };
```

كدا لسة
محجزتش حاجة
في الـ memorey

```
int main ( )
 { struct Student Mohamed , Ali;

 }
```

كدة
حجزت

Cont. assignValue ~~definition~~ ① اول طريقة لل

```
Mohamed.Class = 1;
Mohamed.Id    = 1033;
Mohamed.grade = 'A';
Ali.Class     = 3;
Ali.Id        = 725;
Ali.grade     = 'B';
```

Dot operator ⇒ struct | → arrow operator ⇒ ptr to struct

---

② تاني طريقة لل definition

```
struct Student Mohamed = {1, 1033, 'A'};
```

③ تالت طريقة ال definition

```
struct Student Mohamed = {.Id=1033};
```

---

④ رابع طريقة لل defined

```
struct Student
{
    int seat;
    char name;
    float grad;
} Mohamed = {2, 'A', 3.2}, Ali = {4, 'B', 4.7};
```

---

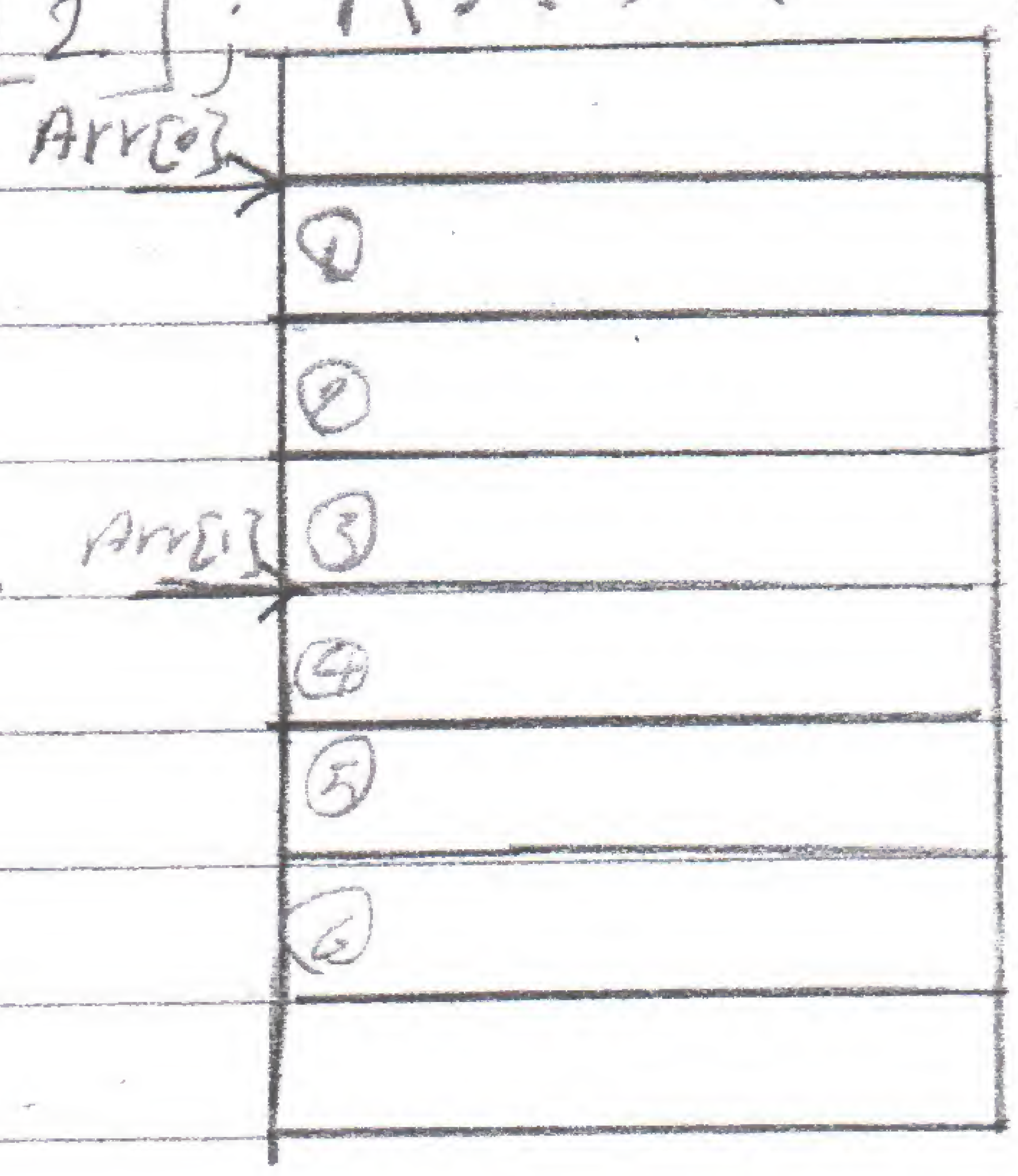
* آل Body بتاع ال Struct عبارة عنه
السبلونة أو الاسطمبة Stamp مش بتحجز حاجة
في ال memory . مش بتحجز غير لما ت Create منها

Array of struct

```
main( )
{       struct student  Arr[2];

        Arr[0]. Class = 2 ;
        Arr [0]. grad= 3.4;
        Arr[0]. Nam = 'A';
        Arr [1] . Class = 6 ;
        Arr [1] . grad = 7.3 ;
        Arr[1] . Name = 'B';
}
```

RAM



---

[11] Code EX 11:

write C code of Array of struct scan it from user and print it.

Type def

```
Typedef struct
        { int seet;
          char Name;
        } Student;
```

Not optinal

* كلما يعرفوا إنها بتشيل ال Struct لما تيجي ؛ Creat X

ب Creat ← New Type

Pointer to array of struct.
[illegible]

أول ما تسمع
الاسم ده أول
→ حاجة تيجي على بالك

Pointer to struct

struct student Ali;
struct student *ptr = &Ali;
ptr -> class = 2;
ptr -> grad = 4.3;
Arrow operator

ptr.class X
ptr->class ✓
ptr[3].class ✓
ptr[3]->class X

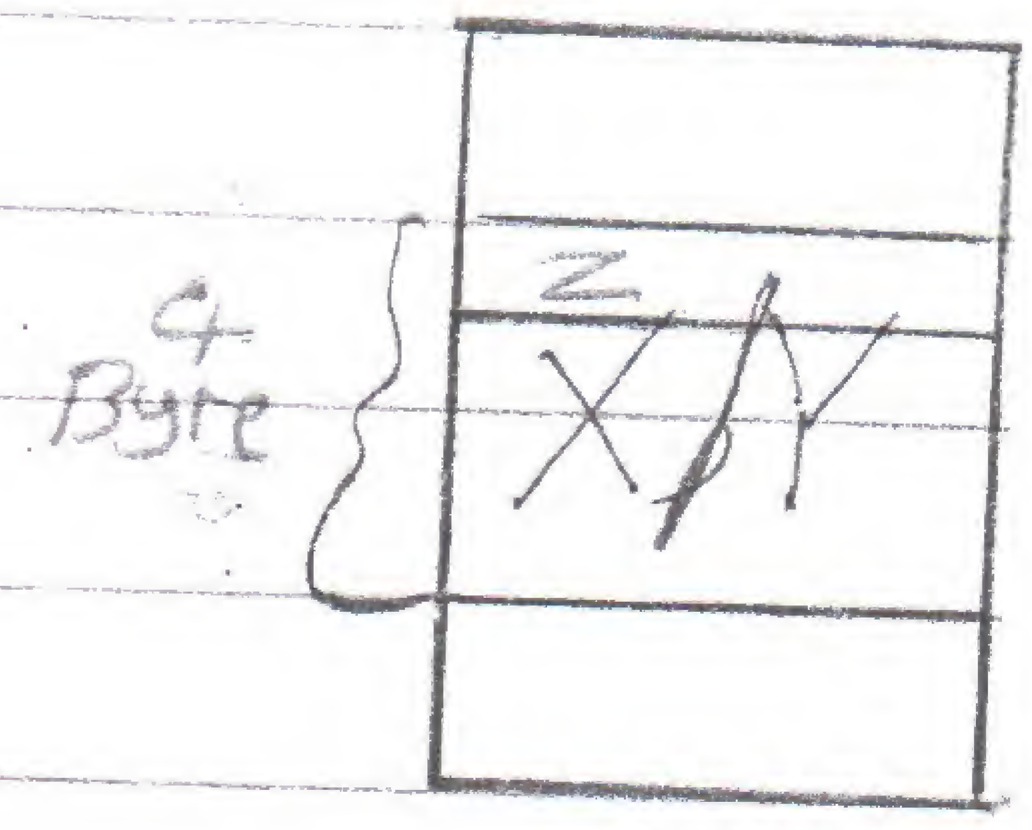
---

12 Code EX12:
Write C Code of function that takes struct by address and print it;

Union

* exact like struct. it reserve the biggest data type in it.

```
Union test
{ int x;
  int y;
  Char z;
};
```



2012/2016

Exponat & mantissa
الاس (مئوس) الكسر

float x = 10.375;

Sign	exponant	mantissa

1 Bit — 8 Bit — 23 Byte

4 Byte
32 Bit

10.375

(Binary) 1010

0.375	x2	0.75	0
0.75	x2	1.5	1
0.5	x2	1	
0			

1010.011

Binary Base

3 + 127 = 130 → exponat

1.010011 * 2^3

Sign	exponent	mantissa
0	130 → 10000010	01001100000000000000000

N.B if The divison is endless
∴ Float > double → in binary.

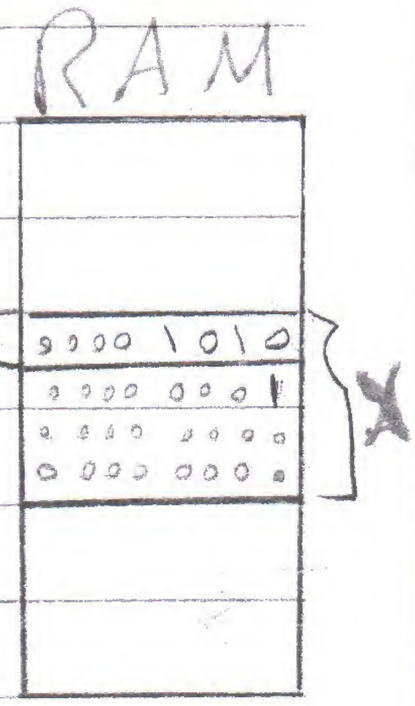
2.75 → 421 • 1/2 1/4 1/8
010.110

00

N.13

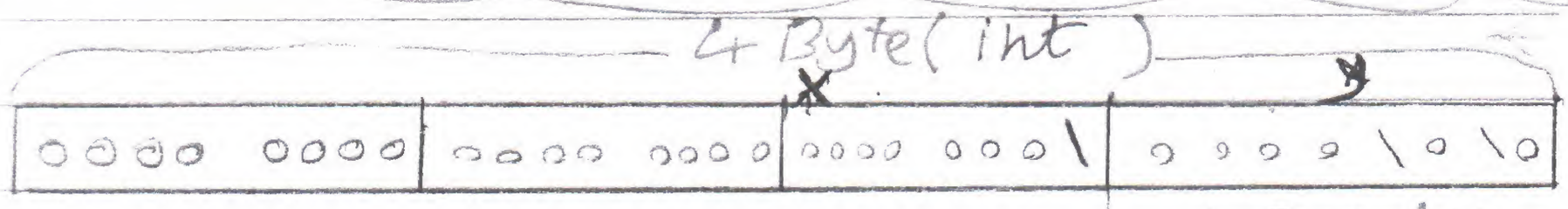
* اى حاجة متعرفة Global بـ initialize بـ Zero .BSS
* اى حاجة متعرفة Local بـ initialize بـ Rubbish
* بيتعمل اساوى Struct بـ Struct (بيساوى قيمهم)
* لما أقلك Pointer to struct اول حاجة تيجى فى دماغك ←
* Pointer before access
* الفرقه بين → و •

Union




```
union test
  { char y;
    int x;
  }
int main ( )
  { union test My_Test;
    My_Test.x = 266;
    printf ("%d", My_Test.y); → 10
  }
```

IT IS NOT CYCLEC Property



* لو عكست وحطيت قيمة 257 y وطبعت الـ x
هتعمل Cyclee Prop

enum

```
                0    1    2    3  =12  4  =13  5  =14  6  =15
enum week { Sat, sun, mon, Tus, wed, Thr, fri };
int main( )
{   enum week Today ;
    Today = sat ;
    printf("%d", today);
}
```

* لو عرفتلة اى element مثلاً "Sat=17" هيكمل لوحده بتسلسل san=18, mon=19 , Thar=20 ...

"مهم جداً جداً" Boolean STring

* مفيش Boolean في الـ C-language بس ممكن نعمله بالـ enum

⇉ enum Boolean { fals, True };

---

* مفيش STring في الـ C-language بس ممكن اعملها بالـ arr of char

⇉ char arr[ ] = "Mohamed";
   printf("%s", arr);

* او ممكن نعملها كدة بس من غير ما تحجز مكان في الـ memory

⇉ char *ptr = "Mohamed";
   printf("%s", ptr);

DATA STRUCTURE

Sorting

1) Bubble_Sort.

2 for Loops "Nested"

Arr[10] = {12, 11, 13, 14, 2, 4, 9, 19, 15, 10}

```
iF (Arr[i] > Arr[i+1])
swap(Arr[i], Arr[i+1]);
```

	0	1	2	3	4	5	6	7	8	9
1-(0,1)	{11,	12,	13,	14,	2,	4,	0,	19,	15,	10}
2-(1,2)	{11,	12,	13,	14,	2,	4,	0,	19,	15,	10}
3-(2,3)	{11,	12,	13,	14,	2,	4,	0,	19,	15,	10}
4-(3,4)	{11,	12,	13,	14,	2,	4,	0,	19,	15,	10}
5-(4,5)	{11,	12,	13,	14,	2,	4,	0,	19,	15,	10}
6-(5,6)	{11,	12,	13,	14,	2,	0,	4,	19,	15,	10}
7-(6,7)	{11,	12,	13,	14,	2,	0,	4,	19,	15,	10}
8-(7,8)	{11,	12,	13,	14,	2,	0,	4,	15,	19,	10}
9-(8,9)	{11,	12,	13,	14,	2,	0,	4,	15,	10,	19}

* بعدما لفوا لنهاية ال (size-1) يبقى اكبر رقم في الآخر.
* محتاجين نكرر العملية دي بعدد ال (size) بتاع ال Arr.
* يعني يبقى كدا ال size * size = 10 لفحص المثال اللي فوق.

```
#include<stdio.h>

  int Z, j;
Void swap(int *ptr1, int *ptr2)
 {     int Temp;

       Temp = *ptr1;
      *ptr1 = *ptr2;
      *ptr2 = Temp;
 }
Void bubble_Sort(int* ptr, int size)
 {    int i, c, F;
      for(c=0; c<size; c++)
          {
      F=0;
      For(i=0; i<size-1; i++)
      {
          Z++;
        iF(ptr[i]> ptr[i+1])
          {
              swap(&Ptr[i], &Ptr[i+1]);
          }
        else
          {
              F++;
          }
      }
 }
```

```
        if(F==Size)
        {
        }  break ;
    }

    For( i=0; i<size; i++)
    {
        printf("%d\n", ptr[i] );
    }

    printf("\n\n\n%d ", z) ; //num of itration
}

int main( )
{
    int arr[5] = {3, 6, 1, 7, 2};

    bubble_sort (arr, 5);

}
```

2) Selection_Sort

```
Void swap (int *Ptr1, int *Ptr2)
{
    int temp;
    Temp = *Ptr1;
    *Ptr1 = *Ptr2;
    *Ptr2 = temp;
}
int Get_min_index (int *Ptr, int start, int end)
{
    int i j
    int min_indeed = start;
    for(i = start; i<end ; i++)
    {       if(Ptr[i]<Ptr[min_indeed])
                min_indeed = i ;
    }
    return min_indeed ;
}
Void Selection_Sort (int *ptr, int size)
{   int i ;
    int min_index_value ;
    For ( i =0;i<size; i++)
{ min_indeed_value = get_min_index (ptr, i, size);
    swap(&Ptr[i],&Ptr[min_index_value]);
}
```

```
int main ( )
{   int i j
    int arr [10] ={5,33,16,2,54,0,3,7,8};
Selection_sort (arr,10);
    for(i=0;i<10;i++)
      {
        Printf("%d\n",arr[i]);
      }
}
```

[3] Linear search — Searching

```
arr[10]={1,3,0,7,12,2,5,4,6,8}
For(i=0;i<size;i++)
{
  if(arr[i]==value)
  {   return i;
  }
}
```

4 Binary_Search

* لازم تكون مترتبة *

* what is The recursive fanction ?
it is fanction call itself.

```
int binary_Search(int* Ptr, int Low, int high, int Value)
{   int mid = (high + Low) /2;
    if (Ptr [mid] == Value)
    {   return mid;
    }
    else if (Ptr[mid] > Value)
    {
    return binary_search (Ptr, Low, mid-1, Value)
    }
    else
    {   return binery serch (Ptr, mid+1, high, Value)
    }
}
int main ( )
{   int X , result ;
    int arr[10] = {1, 3, 4, 5, 6, 7, 9, 10, 11, 20};
    printf("enter number\n");
    scanf ("%d", &X)
    result = binary_search (arr, 1, 10, X );
    printf ("%d\n", result );
}
```

ال return مترجع واحد قواحدة لغاية ماتدي لأول واحدة

PAGE

DATE

[5] Stack

```
int arr[10];
int c = 0;
void push(int x)
{       if (c != 9)
        {   arr[c] = x;
            c++;
        }
        else
        {   printf("stack is full\n");
        }
}
int pop (void)
{   c--;
    if (c == -1)
    {   printf("stack is empty\n");
        c = 0;
    }
    else
    {       return arr[c];
    }
int main ( )
{   int x;
    push (44);
    push (60);
    push (30);
    x = pop( );
    printf("%d\n", x);
    push (7);
    push (9);
    x = pop( );
    printf("%d\n", x);
```

queue

```
int arr[10];
int c=0;
void add(int x)
{
    if(c!=10)
    {
        arr[c]=x;
        c++;
    }
    else
    {
        printf("queue is full\n");
    }
}
int get(void)
{
    int temp=arr[0],i;
    for(i=0;i<c;i++)
    {
        arr[i]=arr[i+1];
    }
    c--;
    return temp;
}
int main()
{
    int x;
    add(5);
    add(4);
    add(9);
    add(12);
    x=get();
    printf("%d\n",x);
    x=get();
    printf("%d\n",x);
    x=get();
    printf("%d\n",x);
}
```

PAGE
DATE

7 Linked List

```
Struct node
{   int key;
    int data;
    struct node *pNext;
};
struct node *pstart;
struct node *plast;
struct node New_node(void)
{  Struct node *pNew;
    pNew = (struct node)malloc(sizeof(struct node));
printf("enter key\n");
scanf("%d", &pNew->key);
printf("enter data\n");
scanf("%d", &pNew->data);
pNew->pNext = Null
retarn pNew;
}
Void add_Last(Void)
{   struct node *pNode = New_Node( );
    if (pstart == Null)
    {   pstart = plast = pNode;
    }
```

```
    else
    {   plast->pNext = pNode;
        plast = pNode;
        plast->pNext = null;
    }
}
Void display_all (void)
{   struct node *pdisplay = pstart;
    while (pdisplay != NULL)
        { printf("%d \n", pdisplay->key);
          printf("%d \n", pdisplay->data);
          pdisplay = pdisplay->pNext;
        }
}
Struct node *Search(int value)
{   Struct node* psearch = pStart;
    While (psearch != NULL)
        if (psearch->key == value)
        {   retarn psearch;
        }
        else
        {   psearch = psearch->pNext;
        }
}
```

```
Void display_node(int value)
    Struct  node *ptr;
    ptr = search(value);
    if (ptr==Null)
    {   printf("not found\n");
    }
    else
    {   printf("%d\n", ptr->data);
    }
}
Void delete(Void)
{   int temp;
    Temp = pstart->pNext;
    free (pstart);
    pstart = temp;
}
int main()
{   int x, y;
    while (1);
    {   printf("1=add new\n 2=printall\n 3=Search 4. delete
        scanf("%d", &x);
        Fflash (Stdin);
        switch (x)
        {   Case 1:
                add_last();
            break;
```

```
case 2:
        display_all();
        break;
case 3:
        printf("enter key to search\n");
        scanf("%d", &y);
        display_node(y);
        fflush(stdin)
        break;
case 4:
        delete();
        break;
default:
        printf("only enter 1, 2, 3 or 4");
        break;
    }
  }
}
```

Dynamic memory allocation

* How to resirve in the life time

1- malloc( );
2- calloc( );
3- free ( );
4- Realloc( );

RESERVE IN HEAP

1 malloc

Void* malloc (# of Bytes);

Ptr = (struct*) malloc (size of(struct));

* explecit casting.

* Func بتاخد عدد ال Bytes و بترجع Void Pointer يشاور على أول
اللى أنا حجزته.

2 calloc

Void* calloc (#of element, # of Bytes);

* الفرقه بين ال malloc وال calloc

① ال malloc بتاخد #of Bytes بس ، لكن ال calloc تاخد #of Bytes & #of elements

② ال malloc بتحط garbage اللى بتحجزه ، لكن ال calloc بتحط Zeros

3 Free

free(Ptr);

بدلف ال Ptr اللى أدهولى ال malloc أو calloc

4 realloc

وهى بتشيل الحجر
بتعدل على الحجم القديم

realloc(Ptr, الحجم الجديد size)